Microservices Patterns

micro-23

Aliaksei Bialiauski

Designed in LaTeX

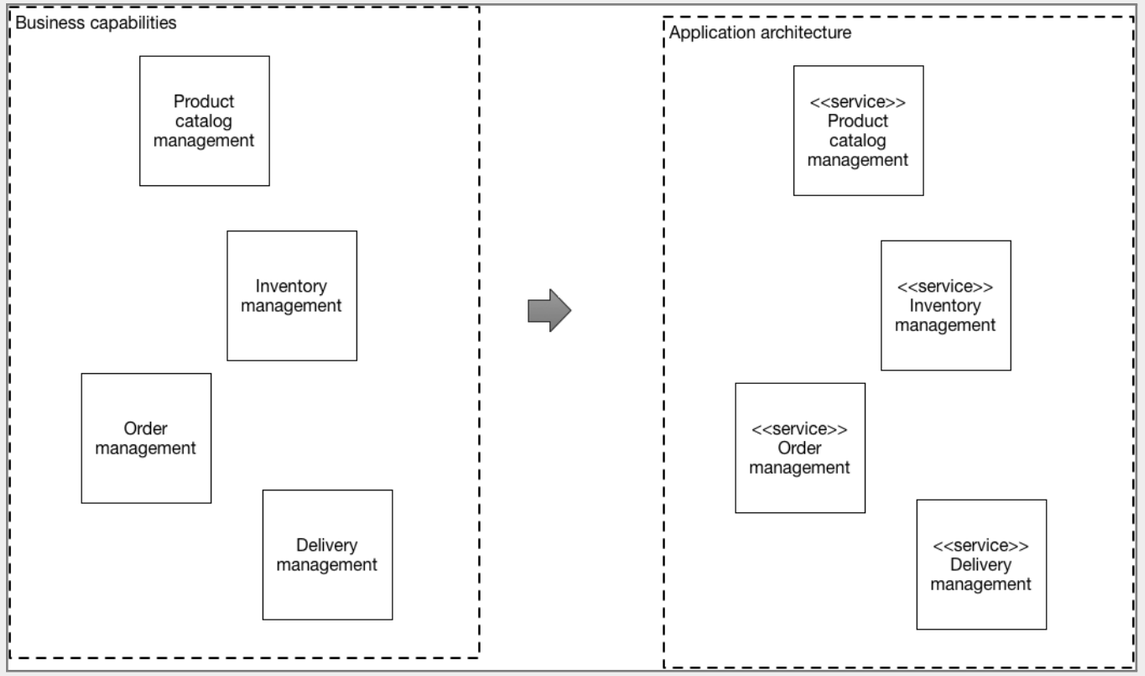
All visual and text materials presented in this slidedeck are either originally made by the author or taken from public Internet sources, such as website. Copyright belongs to their respected authors.

- API Composition
- Service Discovery
- Data Consistency
- Transactional Management
- Reliability
- Observability

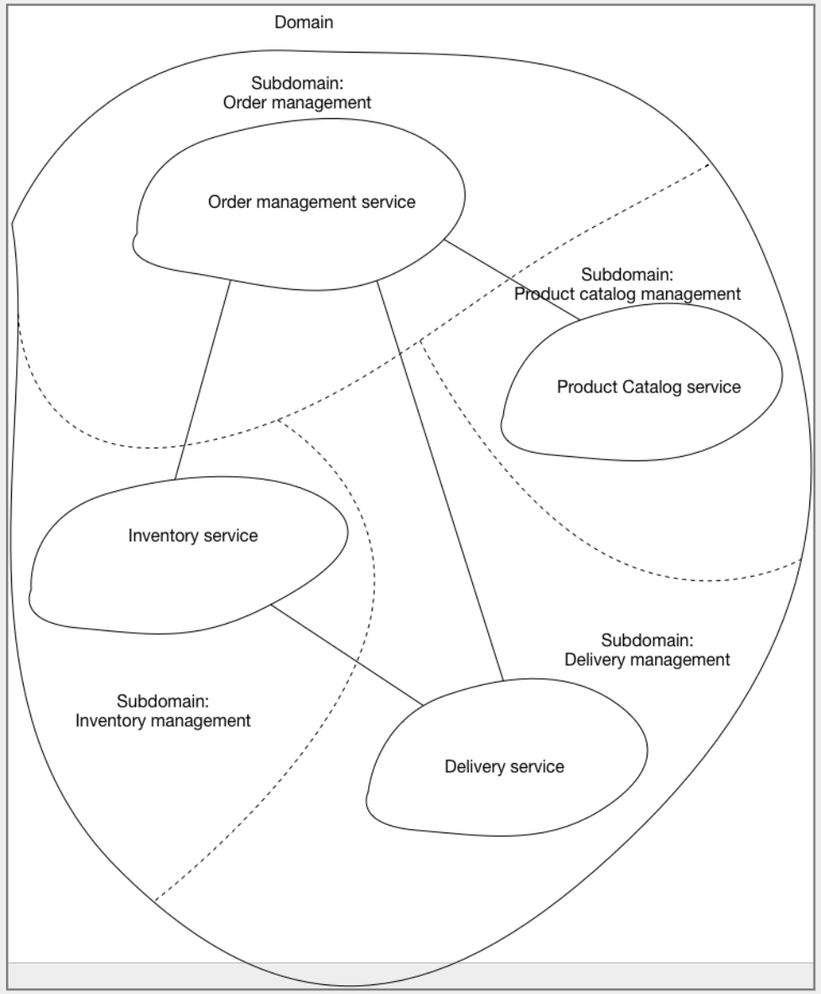
Chapter #1:

API Composition

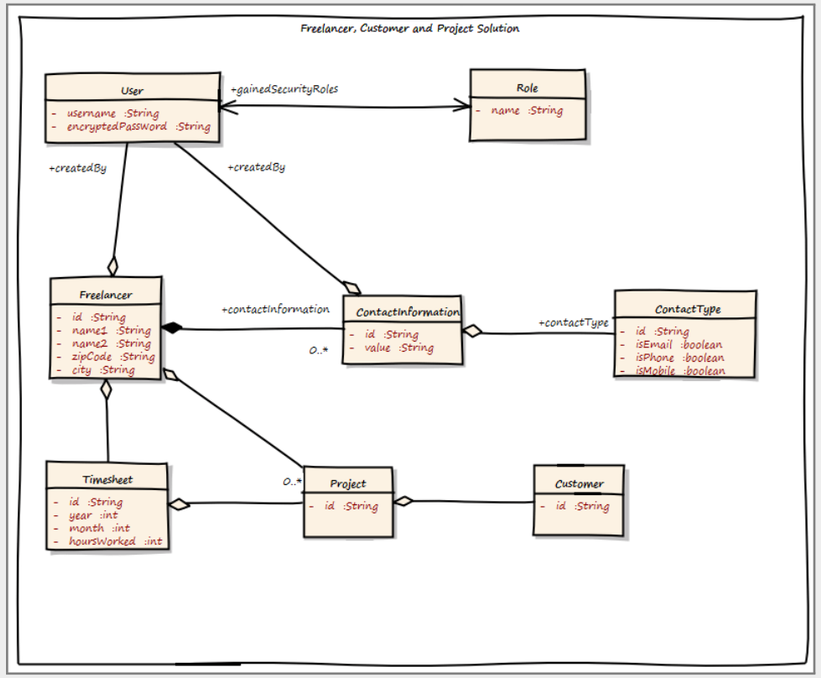
Decompose by Business Capability



Decompose by subdomain



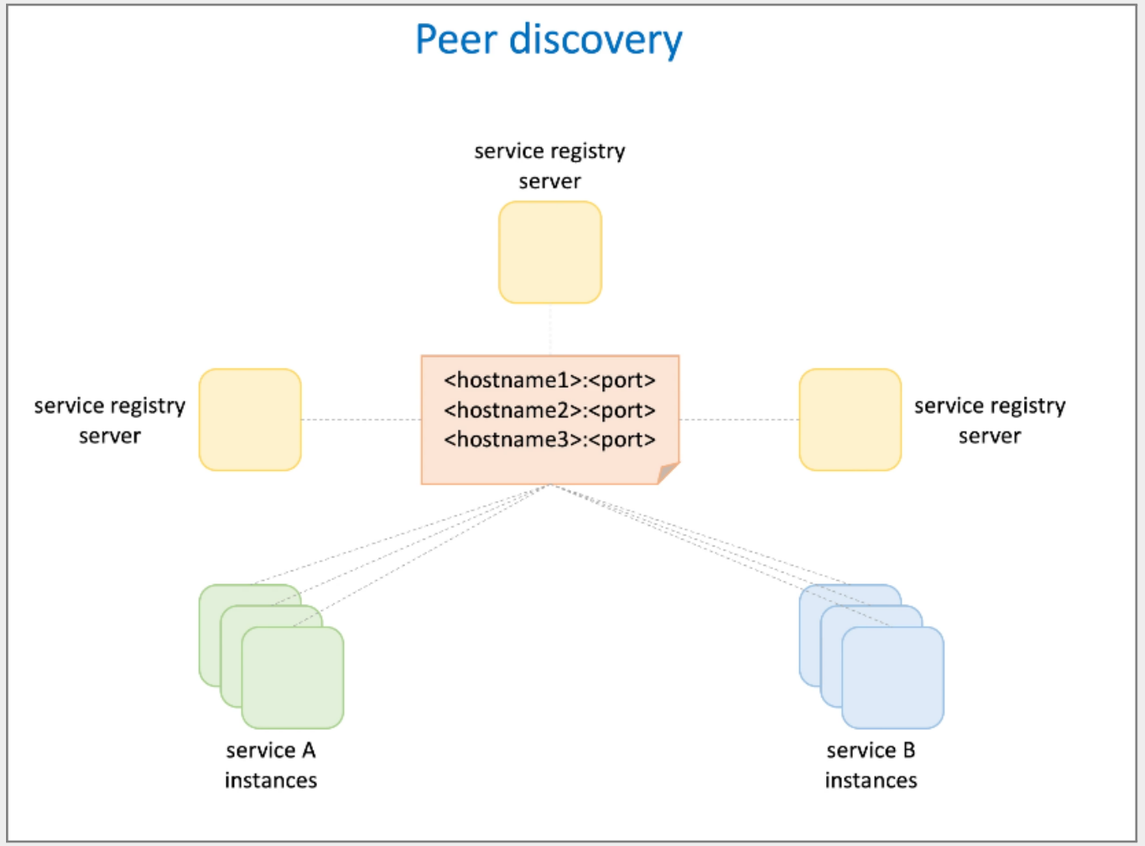
Domain Driven Design



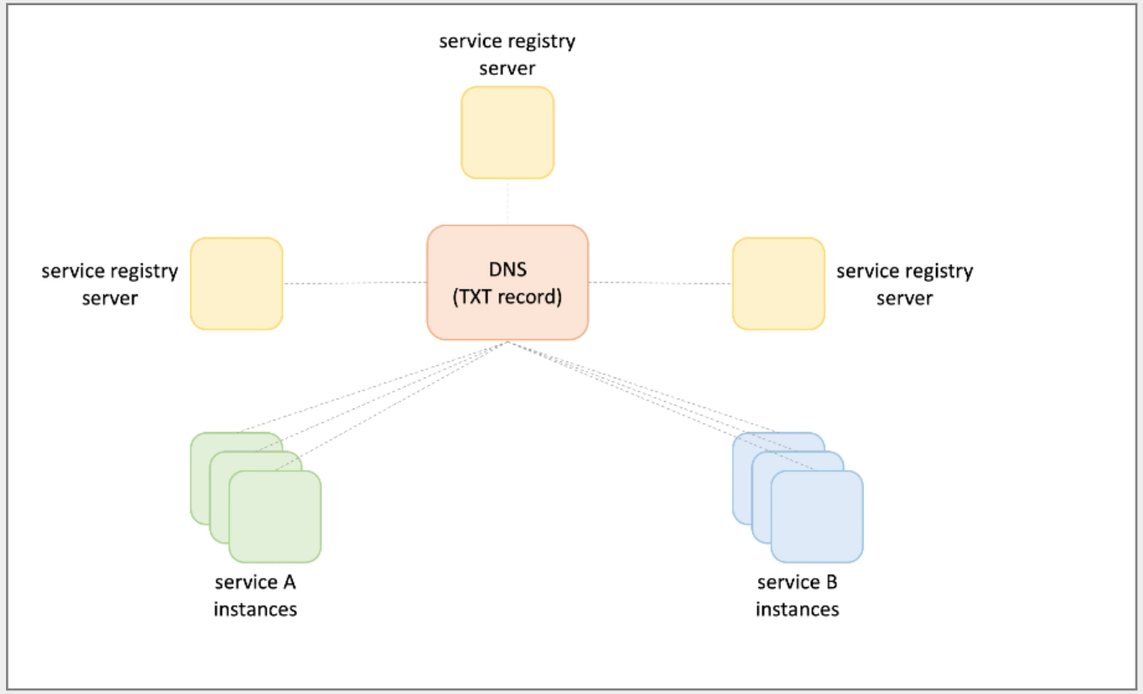
Chapter #2:

Service Discovery

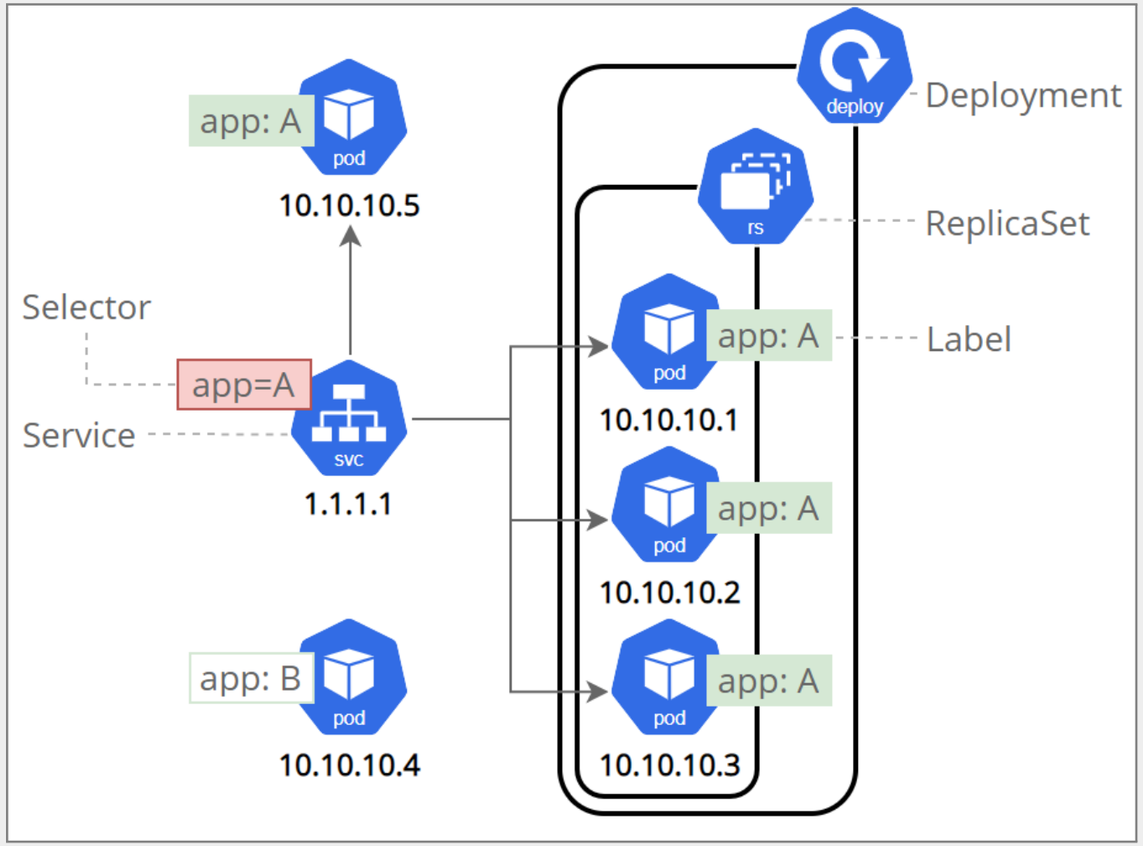
Apache Zookeeper



Spring Cloud Eureka



Kubernetes service registry



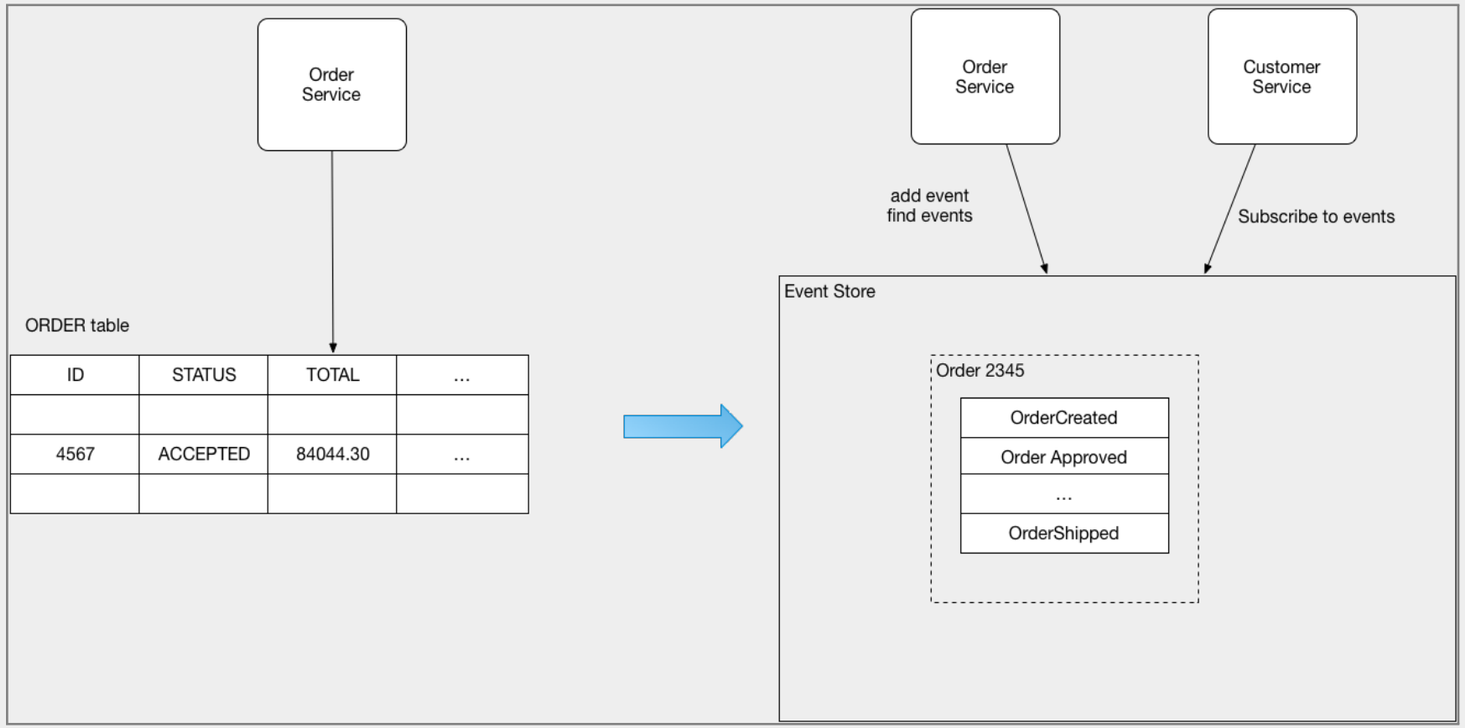
Self registration vs 3rd party registration

"A service instance is responsible for registering itself with the service registry. While a 3rd party registrar is responsible for registering and unregistering a service instance with the service registry."

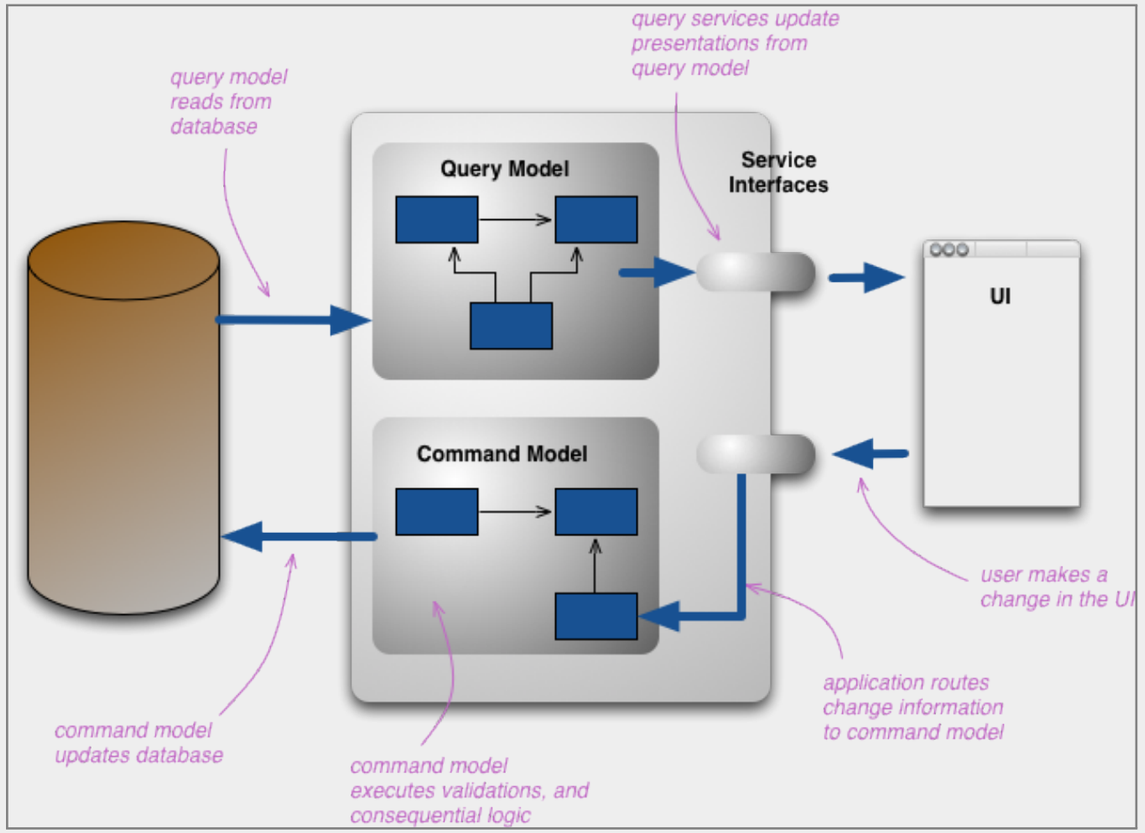
Chapter #3:

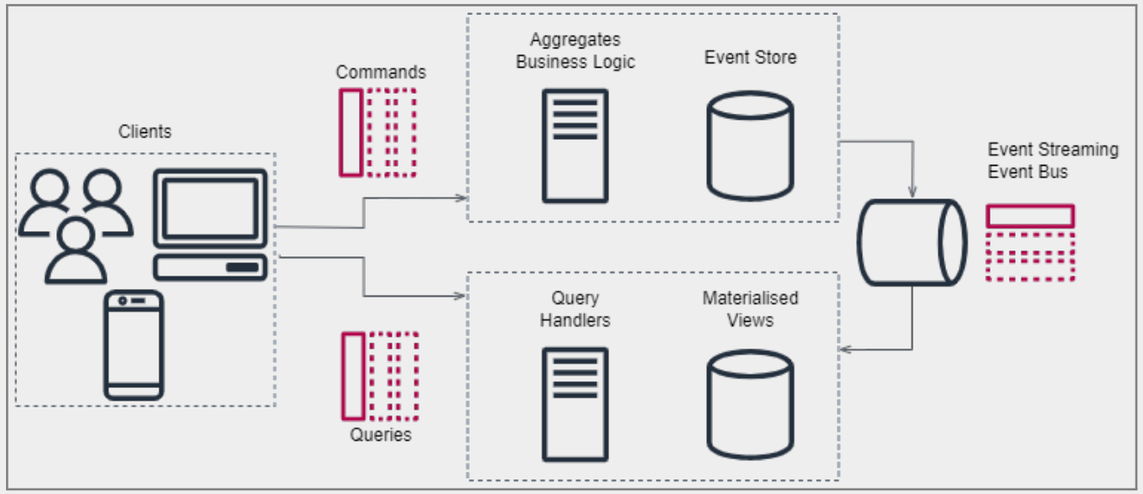
Data Consistency

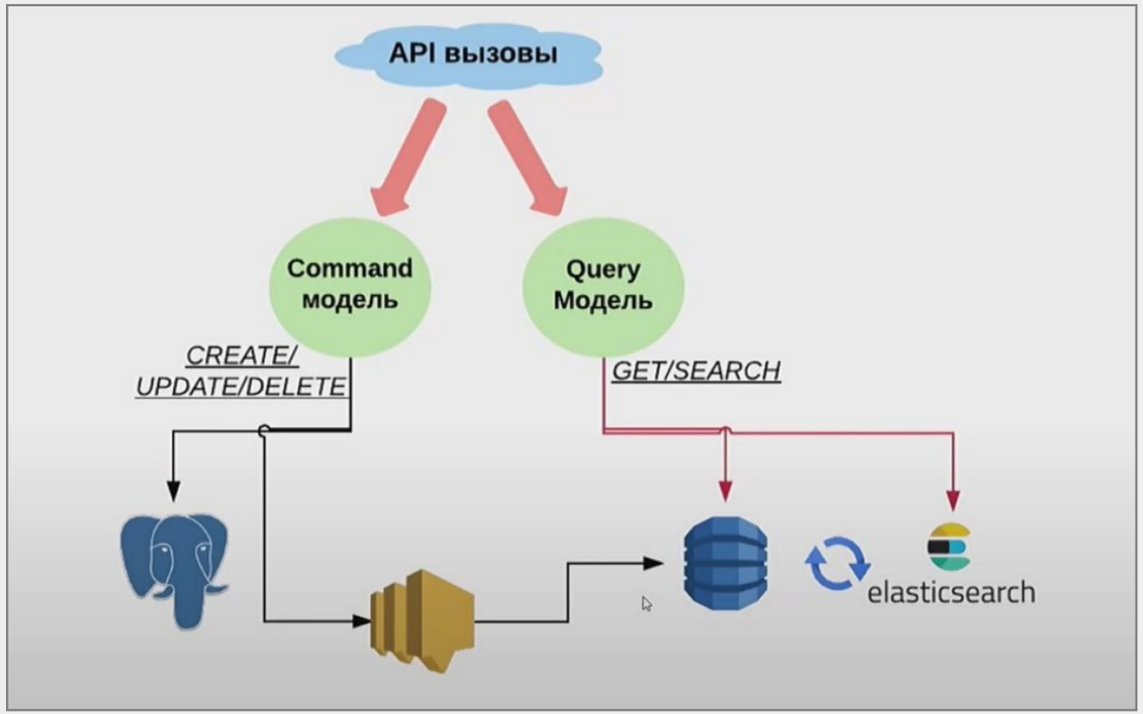
Event sourcing



Command Query Responsibility Segregation



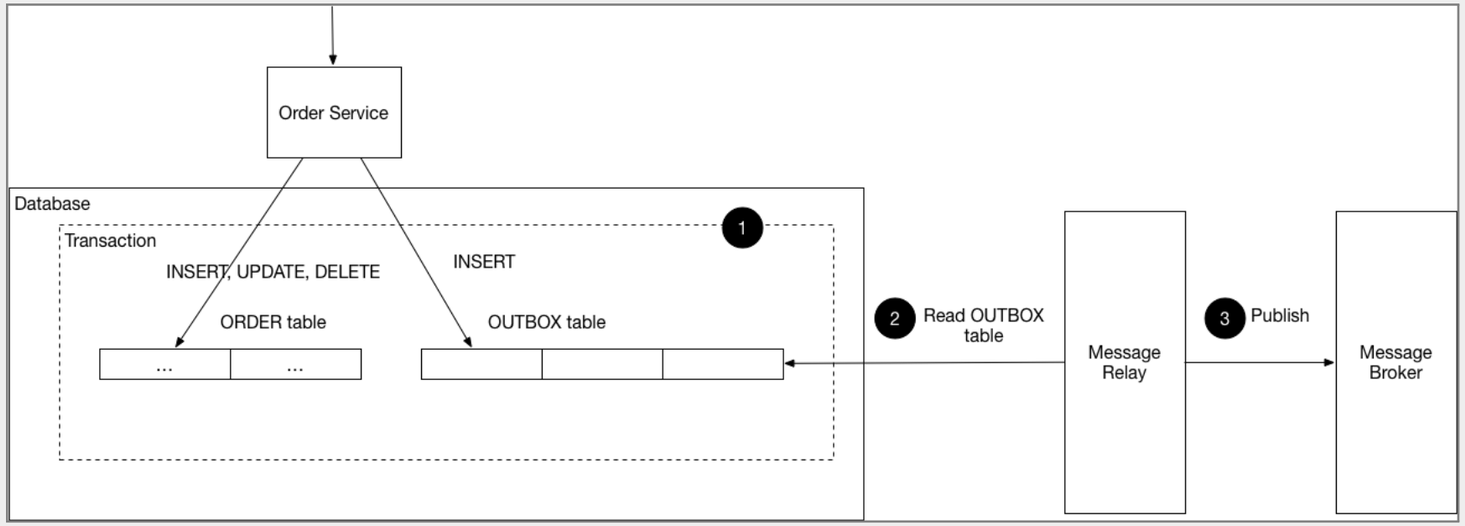

Clients
Commands
Aggregates
Business Logic
Event Store
Event Streaming
Event Bus
Query
Handlers
Materialised
Views
Queries


API вызовы
Command модель
Query Модель
CREATE/ UPDATE/DELETE
GET/SEARCH
elasticsearch

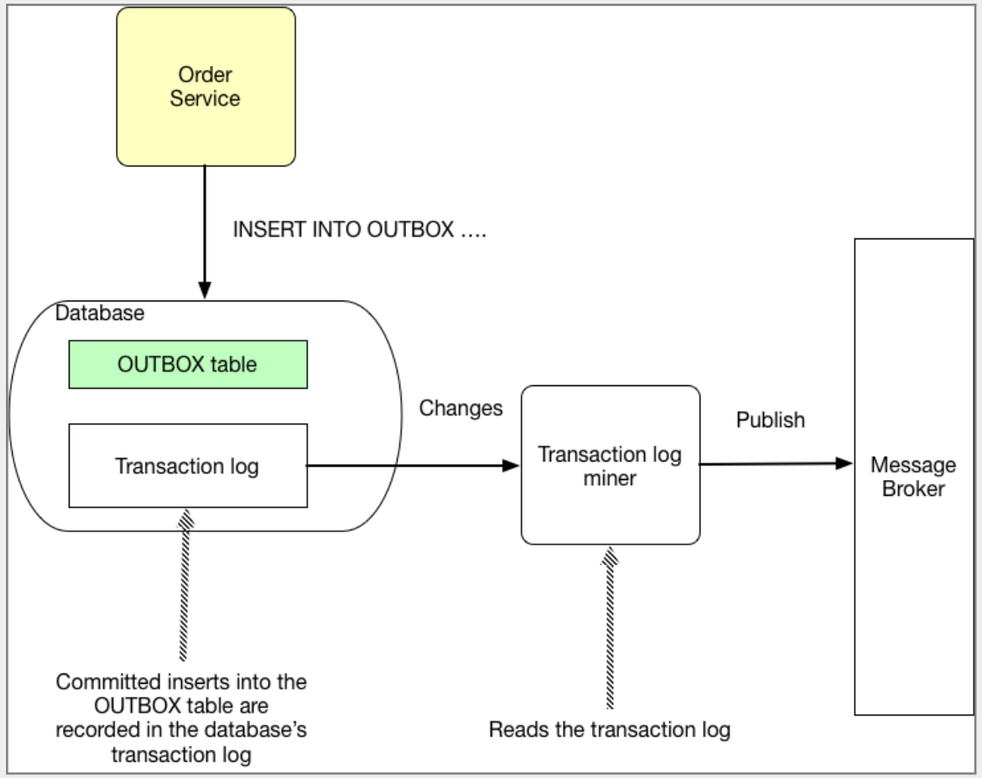
Chapter #4:

Transactional Management

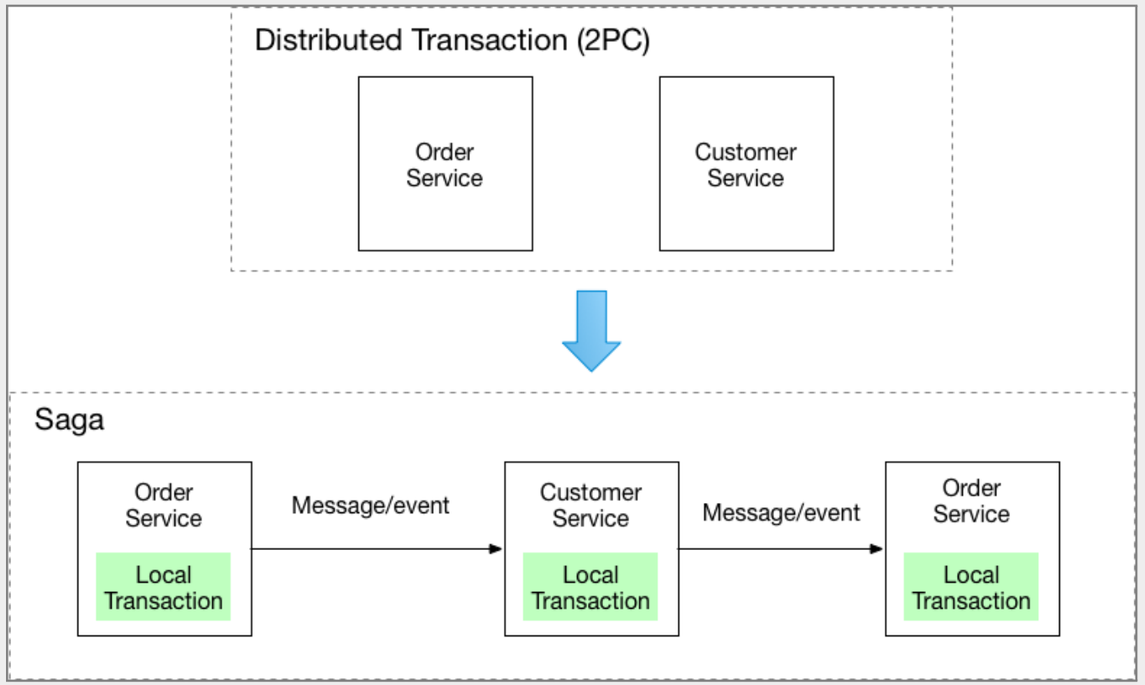
Transactional Outbox



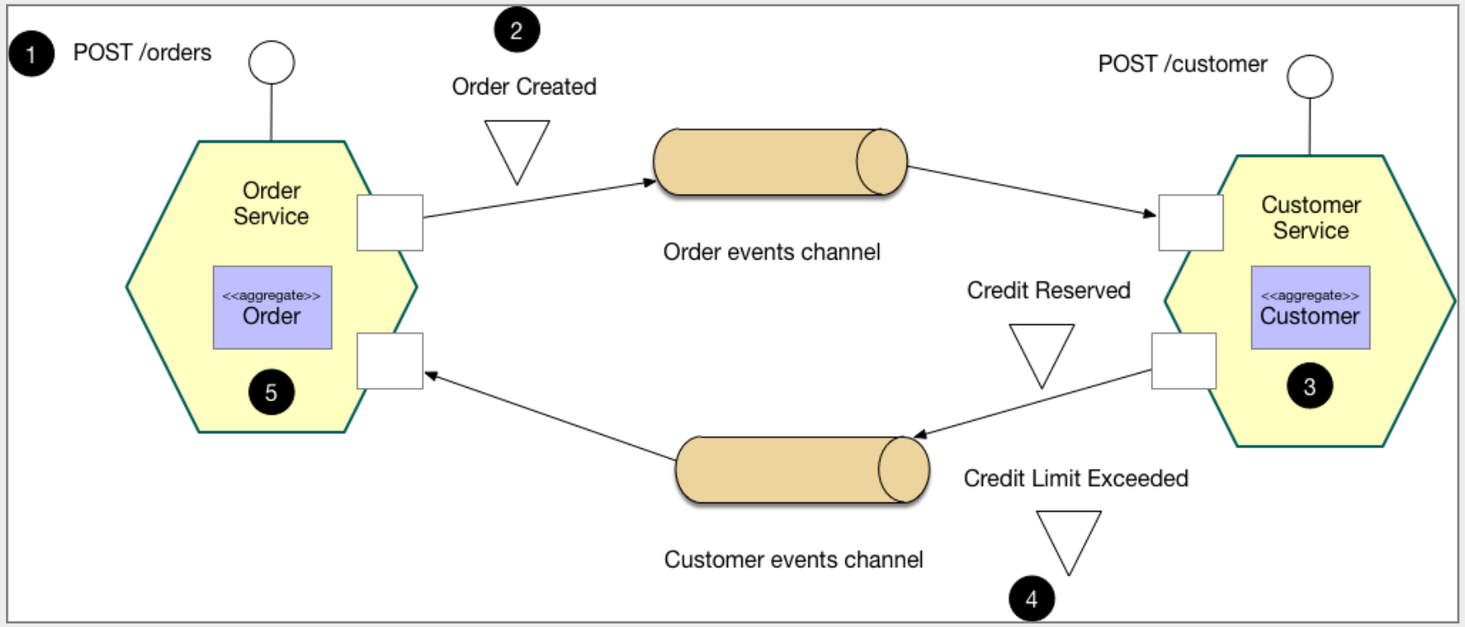
Log tailing


Order Service
INSERT INTO OUTBOX ....
Database
OUTBOX table
Transaction log
Changes
Transaction log miner
Publish
Message Broker
Committed inserts into the OUTBOX table are recorded in the database's transaction log
Reads the transaction log

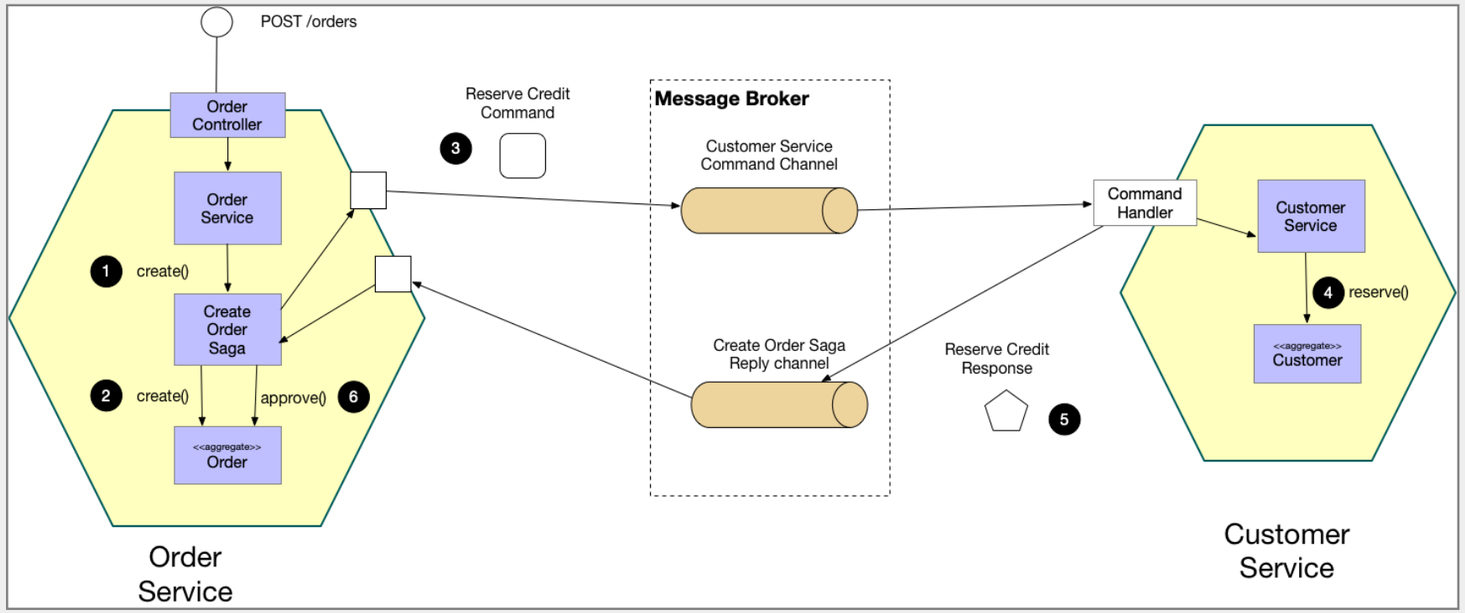
2 Phase Commit



Choreography-based saga



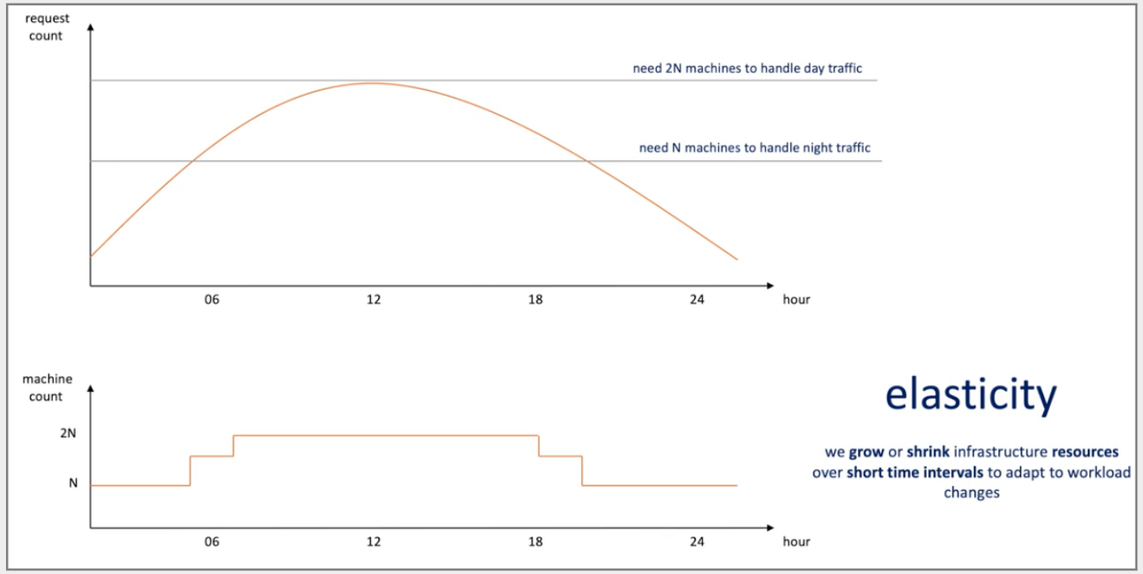
Orchestration-based saga



Chapter #5:

Reliability

Autoscaling



Takes time to provision recources

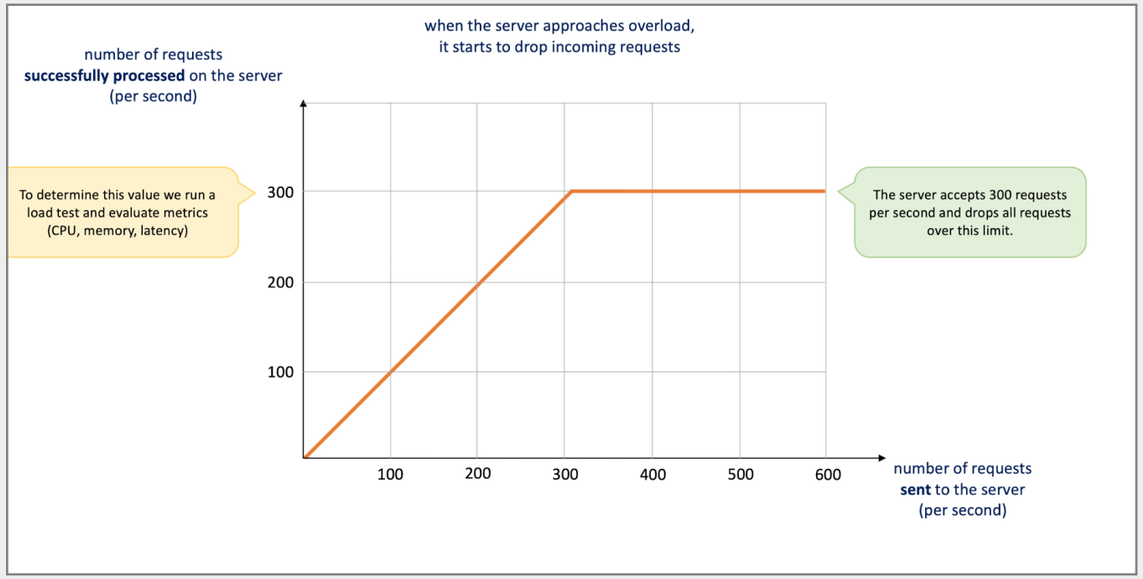
Autoscaling policies

1. metric-based
2. schedule-based
3. predictive

Example

1. CPU util., Mem, Disk
2. Day/Night, business hours
3. CPU increase/decrease

Load shedding



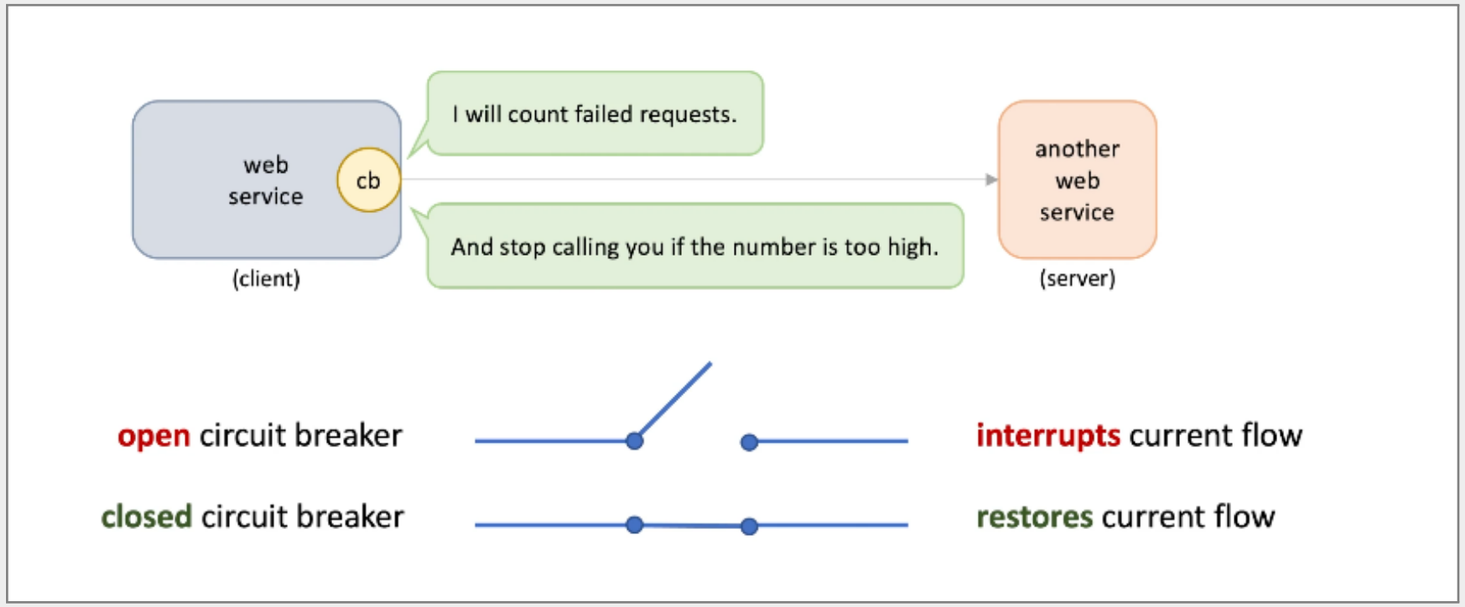
Relys on the state of the system to make the decision

Rate-limiting

https://aws.amazon.com/builders-library/fairness-in-multi-tenant-systems

https://yahooeng.tumblr.com/post/111288877956/
cloud-bouncer-distributed-rate-limiting-at-yahoo

Circuit breaker



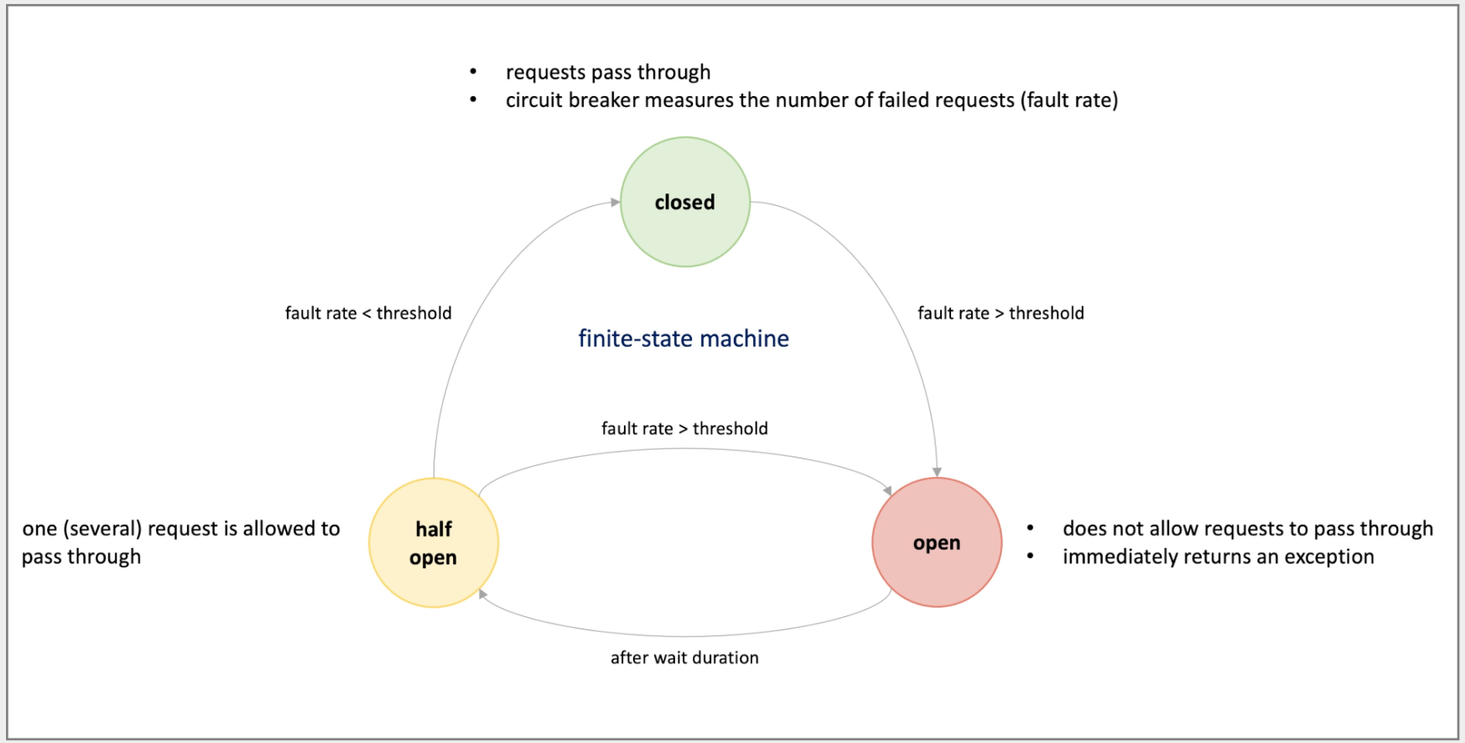
We Configure

- Exception type
- Number of exceptions before circuit breaker opens
- How much time the circuit breaker remains open

Example

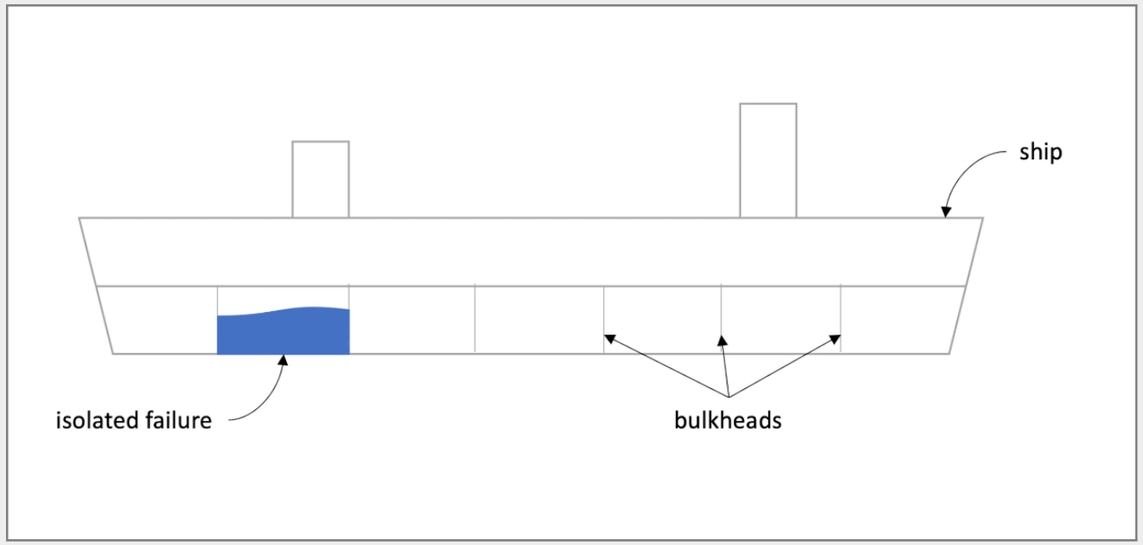
- TooManyRqException
- 5
- 1 min

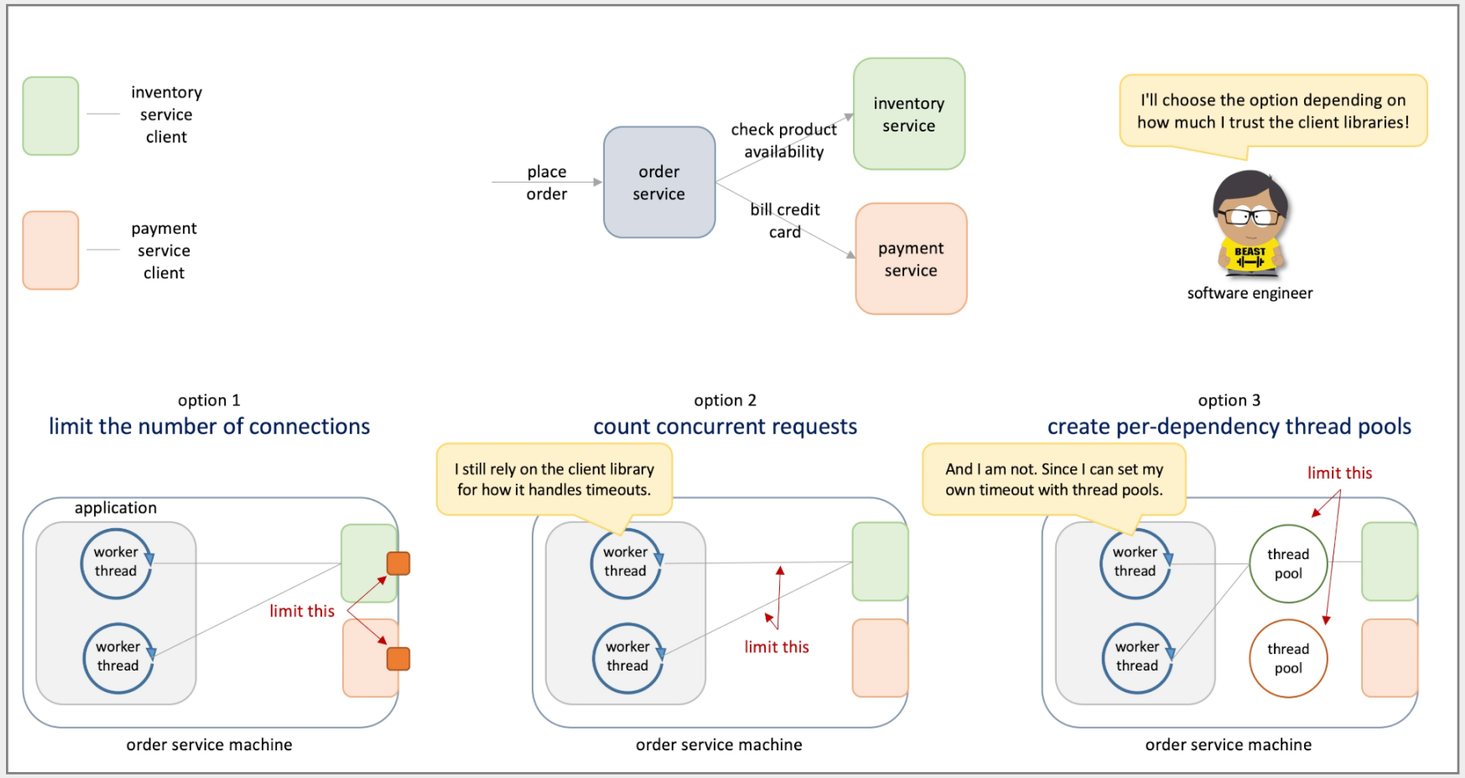
https://resilience4j.readme.io


requests pass through
circuit breaker measures the number of failed requests (fault rate)
closed
fault rate < threshold
fault rate > threshold
finite-state machine
fault rate > threshold
one (several) request is allowed to pass through
half open
open
does not allow requests to pass through
immediately returns an exception
after wait duration

Bulkhead

Partition resources into groups of limited size and isolated groups

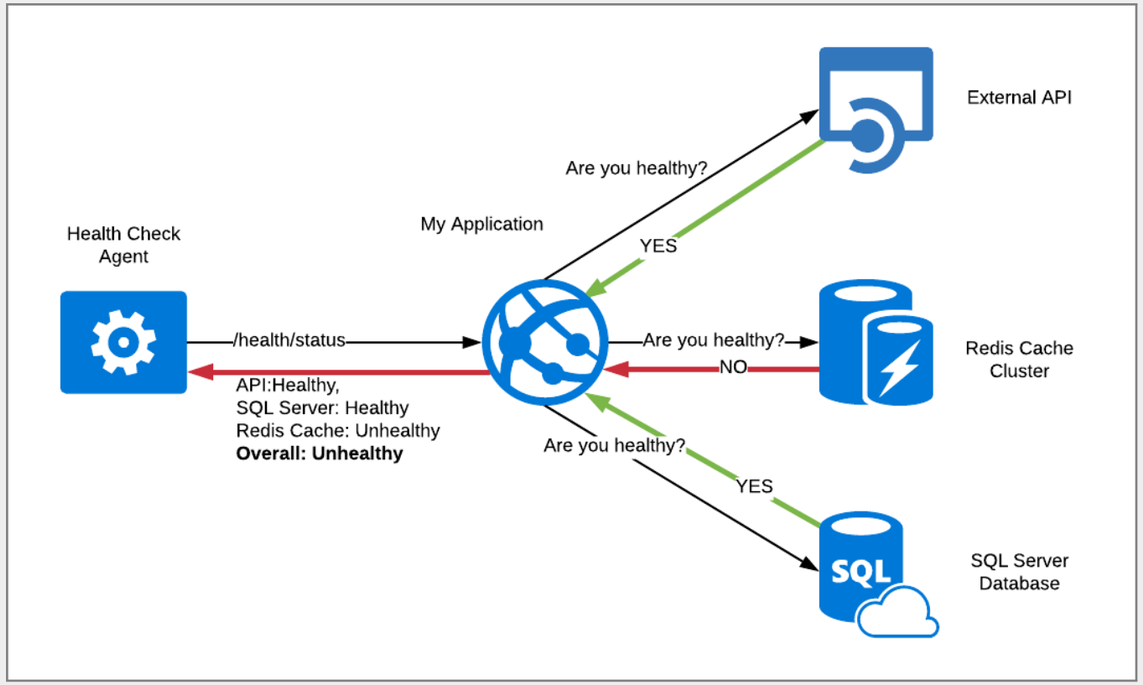



inventory service client
payment service client
place order
order service
check product availability
bill credit card
inventory service
payment service
I'll choose the option depending on how much I trust the client libraries!
BEAST
software engineer
option 1
limit the number of connections
application
worker thread
worker thread
limit this
order service machine
option 2
count concurrent requests
I still rely on the client library for how it handles timeouts.
worker thread
worker thread
limit this
order service machine
option 3
create per-dependency thread pools
And I am not. Since I can set my own timeout with thread pools.
limit this
worker thread
worker thread
thread pool
thread pool
order service machine

Chapter #6:

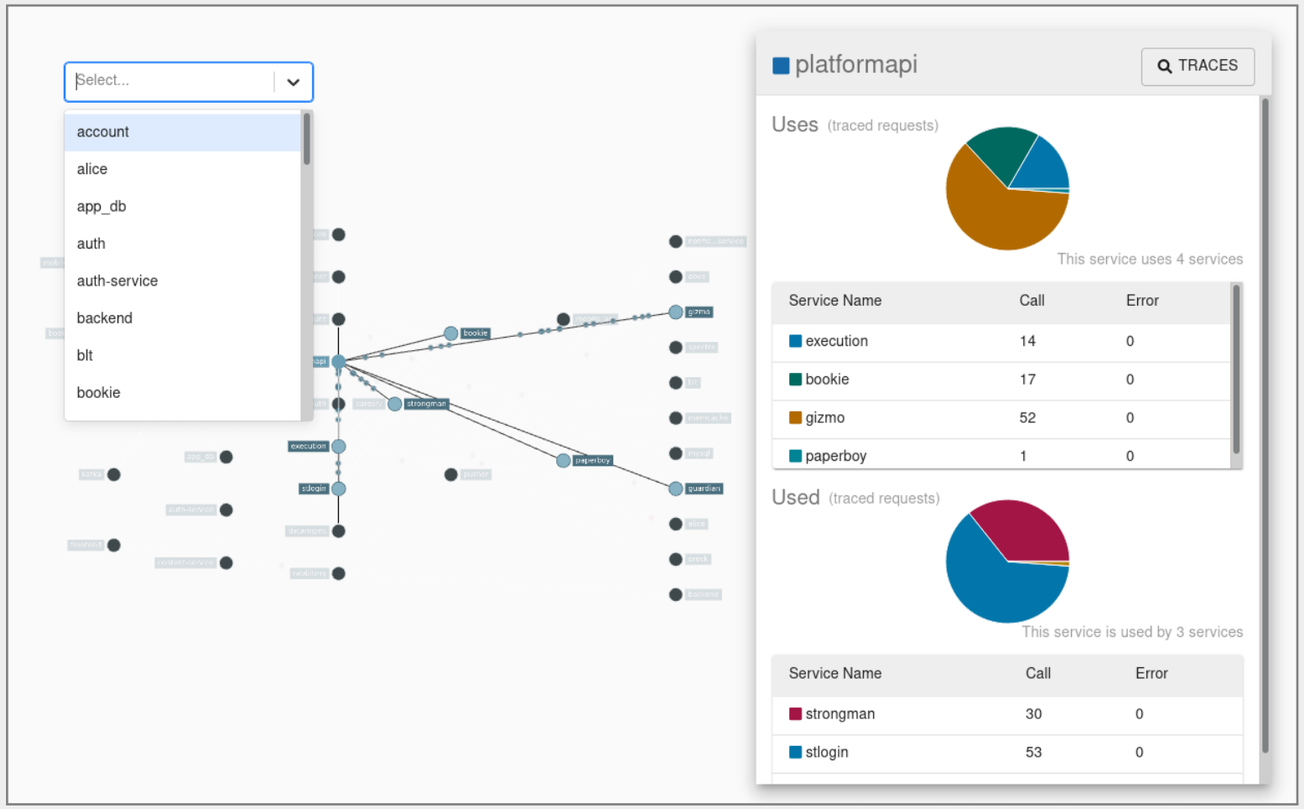
Observability

Health checks




```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-actuator</artifactId>
</dependency>
```

Distributed Tracing



Centralized Logging

Centralized logging service is a service that aggregates logs from each service instance. The users can search and analyze the logs. They can configure alerts that are triggered when certain messages appear in the logs.

https://aws.amazon.com/cloudwatch/

https://www.elastic.co/what-is/elk-stack

Collect and store exceptions

https://sentry.io/